

JP2002091780 A

SYSTEM AND METHOD FOR PROCESSING TRANSACTION

NEC CORP

Inventor(s):HATANO YOJI

Application No. 2000276347 JP2000276347 JP, Filed 20000912,A1 Published  
20020329Published 20020329

Abstract:

**PROBLEM TO BE SOLVED:** To provide a method and a system for processing transaction, by which operation management or transaction processing program can be easily customized by user and a client application can be easily prepared.

**SOLUTION:** The client/server type transaction processing system capable of online transaction processing, has a storage device for storing plural work components as a program for executing transaction processing, a work flow method as a program for calling and executing the work components and a work flow class as a program for storing the work flow method required for every work processing in a server node and a processor for calling the work component through the work flow method for the unit of work flow class and executing transaction processing according to this work component.

Int'l Class: G06F00946; G06F00944 G06F01200 G06F01516

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-91780

(P2002-91780A)

(43) 公開日 平成14年3月29日 (2002.3.29)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テームト <sup>*</sup> (参考)
G 0 6 F 9/46	3 4 0	G 0 6 F 9/46	3 4 0 B 5 B 0 4 5
9/44		9/44	5 3 0 P 5 B 0 7 6
	5 3 0	12/00	5 1 8 A 5 B 0 8 2
12/00	5 1 8	15/16	6 2 0 F 5 B 0 9 8
15/16	6 2 0		6 2 0 T

審査請求 有 請求項の数 6 O L (全 13 頁) 最終頁に続く

(21) 出願番号 特願2000-276347 (P2000-276347)

(22) 出願日 平成12年9月12日 (2000.9.12)

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 波多野 陽治

東京都港区芝五丁目7番1号 日本電気株式会社内

(74) 代理人 100088328

弁理士 金田 暢之 (外2名)

Fターム(参考) 5B045 BB11 BB28 BB47 GG09

5B076 DB04 DC09 DD10

5B082 AA01 GB00

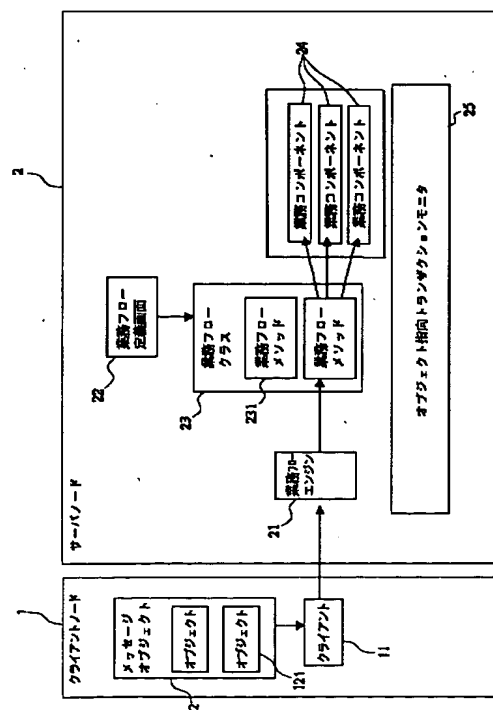
5B098 AA10 GA01 GC01 GC16 GD02  
GD14

(54) 【発明の名称】 トランザクション処理システム及び方法

(57) 【要約】

【課題】 ユーザによる運用管理やトランザクション処理プログラムのカスタマイズが容易であり、クライアントアプリケーションを簡易に作成できるトランザクション処理方法及びシステムを提供する。

【解決手段】 オンラインでトランザクション処理が可能なクライアント/サーバ型のトランザクション処理システムであって、サーバノードに、トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な業務フローメソッドを格納するためのプログラムである業務フロークラスを格納する記憶装置と、業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行する処理装置とを有する構成とする。



## 【特許請求の範囲】

【請求項1】 クライアントとして動作するクライアントノードと、

所定のトランザクション処理を実行するサーバノードとを有し、オンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムであって、

前記サーバノードは、

前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスをそれぞれ格納する記憶装置と、

前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する処理装置と、を有するトランザクション処理システム。

【請求項2】 前記サーバノードは、

前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示するための出力装置を有する請求項1記載のトランザクション処理システム。

【請求項3】 前記クライアントノードは、

前記サーバノードに業務処理を依頼するためのアプリケーションであるクライアントと、

前記クライアントノードから前記サーバノードに業務処理を依頼する際に前記業務コンポーネントに渡すオブジェクトを格納するためのプログラムであるメッセージオブジェクトと、をそれぞれ格納する記憶装置を有し、

前記メッセージオブジェクトに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとが表形式で格納される請求項1または2記載のトランザクション処理システム。

【請求項4】 クライアントとして動作するクライアントノードと、

所定のトランザクション処理を実行するサーバノードと、を有するクライアント／サーバシステムに実行させるためのトランザクション処理方法であって、

前記サーバノードに、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスを予め格納しておき、前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実

行するトランザクション処理方法。

【請求項5】 前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示する請求項1記載のトランザクション処理方法。

【請求項6】 前記クライアントノードに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとを表形式から成るメッセージオブジェクトとして予め格納しておき、

前記クライアントノードから前記サーバノードに業務処理を依頼する際に、前記メッセージオブジェクトを参照し、前記業務コンポーネントに前記トランザクション処理で用いるオブジェクトを渡す請求項1または2記載のトランザクション処理方法。

## 【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はオンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムに関する。

【0002】

【従来の技術】従来、オープン系のトランザクション処理システムは、トランザクションモニタ上で構築され、トランザクション処理プログラムはトランザクションモニタ上で動作するライブラリにより作成されてきた。

【0003】最近の動向として、トランザクションモニタがオブジェクト指向のコンポーネントの動作環境を提供するようになり、トランザクション処理プログラムは、EJB（Enterprise Java Beans）、COM（Component Object Model）、CORBA（Common Object Request Broker Architecture）等によるコンポーネントとして作成される。こうしたトランザクションモニタを以下ではオブジェクト指向トランザクションモニタと呼ぶ。

【0004】上述したようにコンポーネントを用いてトランザクション処理プログラムを作成すると、プログラムが部品化されるため再利用性を高めることができる。例えば、オブジェクト指向トランザクションモニタ上で作成したEJBコンポーネントは、このオブジェクト指向トランザクションモニタ特有のAPI（Application Program Interface）等を使用していない限り、他のオブジェクト指向トランザクションモニタでも動作させることができる。例えば、BEA Systems社のWebLogic Serverの場合、コンポーネントにJ2EE（Java 2 Enterprise Edition：サンマイクロシステムズ社）仕様で決められたEJBをサポートしているため、EJBでトランザクション処理プログラムを作成することが可能である。これにより、WebShop等のWebブラウザから投入されたトランザクションをWebサーバ上のサーブレットに渡し、サーブ

レットでWebLogic Server上のEJBで作成されたコンポーネント群を呼び出すことによりトランザクション処理を実行するシステム等を構築することができる。

【0005】このようなトランザクション処理プログラムは一般に複数のコンポーネントの呼出し処理により構成される。通常、コンポーネントの粒度は任意であり、コンポーネントは必ずしもユーザにとって扱いやすい大きさであるとは限らない。また、オブジェクト指向トランザクションモニタでネイティブに提供する運用管理ツール等もコンポーネント単位で提供されるため、ユーザにとって運用管理が困難な場合がある。例えば、トランザクション処理プログラムの稼働状況を表示させる場合、従来のオブジェクト指向トランザクションモニタの運用管理ツールでは、コンポーネントのメソッドが1秒当たり何件実行されたか等の情報を表示するだけであるため、ユーザにとってわかりやすいものではない。

【0006】このような問題を解決するため、例えば、特開2000-132394号公報では、オブジェクトを、処理フローを制御する制御オブジェクト、及びデータアクセスのインターフェースを実装するオブジェクト管理等に分類してオブジェクト指向業務システムを構築していく指針が提示されている。この場合、設計段階から所定の指針にしたがって統一した設計を行えば、オブジェクトの粒度、境界が不均一になる等の問題が発生せず、制御オブジェクトをユーザの業務に対応したものにすることができる。

【0007】

【発明が解決しようとする課題】しかしながら、上記特開2000-132394号公報に記載された技術では、既に稼働している業務システムに対して適用することができないという問題がある。特に、近年は外部コンポーネントの流通が見込まれるため、既存のコンポーネントをまとめる技術も必要とされる。

【0008】加えて、制御オブジェクトやオブジェクト管理に分類してトランザクション処理プログラムを作成する場合でも、異常時の終了処理やコンポーネント呼出し前処理／後処理等のように、ユーザが独自の処理をトランザクション処理のフローに容易に追加できるものではなかった。

【0009】また、上述したように、トランザクション処理システムでは、コンポーネントの粒度が任意であり、必ずしも運用するユーザにとってわかりやすい大きさではないため、運用管理が難しいという問題がある。

【0010】さらに、トランザクション処理が複数のコンポーネント呼び出し処理で構成されている場合、コンポーネントの呼び出しツールやコンポーネントの呼び出し結果を他のコンポーネントに渡す際のデータ変換ロジック等をクライアント側で実装しているため、クライアント側のアプリケーションの作成負荷が大きくなるとい

う問題があった。

【0011】本発明は上記したような従来の技術が有する問題点を解決するためになされたものであり、ユーザによる運用管理やトランザクション処理プログラムのカスタマイズが容易であり、クライアントアプリケーションを簡易に作成できるトランザクション処理方法及びシステムを提供することを目的とする。

【0012】

【課題を解決するための手段】上記目的を達成するため本発明のトランザクション処理システムは、クライアントとして動作するクライアントノードと、所定のトランザクション処理を実行するサーバノードとを有し、オンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムであって、前記サーバノードは、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスをそれぞれ格納する記憶装置と、前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する処理装置と、を有する構成である。

【0013】このとき、前記サーバノードは、前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示するための出力装置を有していてもよく、前記クライアントノードは、前記サーバノードに業務処理を依頼するためのアプリケーションであるクライアントと、前記クライアントノードから前記サーバノードに業務処理を依頼する際に前記業務コンポーネントに渡すオブジェクトを格納するためのプログラムであるメッセージオブジェクトと、をそれぞれ格納する記憶装置を有し、前記メッセージオブジェクトに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとが表形式で格納されていてもよい。

【0014】一方、本発明のトランザクション処理方法は、クライアントとして動作するクライアントノードと、所定のトランザクション処理を実行するサーバノードと、を有するクライアント／サーバシステムに実行させるためのトランザクション処理方法であって、前記サーバノードに、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスを予め格納しておき、前記業務フロークラス単位で、前記業務フローメソッドにより前

記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する方法である。

【0015】このとき、前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示してもよく、前記クライアントノードに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとを表形式から成るメッセージオブジェクトとして予め格納しておき、前記クライアントノードから前記サーバノードに業務処理を依頼する際に、前記メッセージオブジェクトを参照し、前記業務コンポーネントに前記トランザクション処理で用いるオブジェクトを渡してもよい。

【0016】上記のようなトランザクション処理システム及び方法では、業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行することで、稼動状況やその他物件管理などの各種運用単位を、ユーザにとって分かりやすい業務単位で行うことができる。

【0017】

【発明の実施の形態】次に本発明について図面を参照して説明する。

【0018】図1は本発明のトランザクション処理システムの一構成例を示すブロック図である。

【0019】図1において、本発明のトランザクション処理システムは、クライアントとして動作するクライアントノード1と、コンポーネントにしたがってトランザクション処理を実行するサーバノード2とを有する構成である。

【0020】クライアントノード1は、例えば、パーソナルコンピュータ等の情報処理装置を用いて構成され、サーバノード2に業務処理を依頼するためのアプリケーションであるクライアント11と、クライアント11からサーバノード2に業務処理を依頼する際に、トランザクション処理を実行する業務コンポーネントに渡されるオブジェクト（コンポーネント）121を格納するためのプログラムであるメッセージオブジェクト12とを備えている。

【0021】メッセージオブジェクト12は、ハッシュテーブル構造を備えており、例えば、プログラム言語がJavaの場合、HashTable Classをオブジェクト化したものが使用される。メッセージオブジェクト12には、所定のキーとそれに対応する値とが格納され、値として任意のオブジェクトを指定することが可能である。ここでは、オブジェクト121が指定される。

【0022】一方、サーバノード2は、例えば、ワークステーション等の情報処理装置によって構成され、トランザクション処理を実行するためのプログラムである複

数の業務コンポーネント24と、ユーザによって定義された業務フローにしたがって業務コンポーネント24を呼び出して実行するためのプログラムである業務フローメソッド231と、業務処理毎に、必要な複数の業務フローメソッド231を格納するためのプログラムである業務フロークラス23と、クライアント11からの要求を受け付け、クライアント11により指定された業務フロークラス23による処理を実行させるためのプログラムである業務フローエンジン21とを備えている。なお、業務コンポーネント24は、実際にはサーバノード2が有するオブジェクト指向トランザクションモニタ25上で実行される。また、図1では業務フロークラス23が1つだけ記載されているが、業務の種類に応じて複数の業務フロークラス23を備えていてもよい。

【0023】業務コンポーネント24は、EJB、CORBA等によりユーザが作成するコンポーネントであり、これらの業務コンポーネントによりトランザクション処理が実行される。すなわち、トランザクション処理はこれら複数の業務コンポーネントの呼出し処理で構成される。

【0024】業務フローエンジンは21は、サーバノード2にコンポーネントとして実装され、業務フローエンジン21のメソッドは、周知のコンポーネント呼出しツールによってクライアント11から呼び出される。

【0025】業務フローは、業務コンポーネント24の呼出し順序を定義したものであり、業務フロー定義画面22を用いてユーザにより登録される。また、この業務フロー定義画面22を用いて登録された業務フローからサーバノード2により業務フロークラス23が構成される。

【0026】クライアントノード1あるいはサーバノード2となる情報処理装置は、例えば、図2に示すように、プログラムにしたがって所定の処理を実行する処理装置110と、処理装置110に対してコマンドや情報等を入力するための入力装置120と、処理装置110の処理結果をモニタするための出力装置130とを有する構成である。

【0027】処理装置110は、CPU111と、CPU111の処理に必要な情報を一時的に記憶する主記憶装置112と、CPU111に本発明の業務システムの処理を実行させるためのプログラムが記録された記録媒体113と、クライアントノード1としての処理やサーバノード2としての処理に必要なプログラムやデータが格納されるデータ蓄積装置114と、主記憶装置112、記録媒体113、及びデータ蓄積装置114とのデータ転送を制御するメモリ制御インタフェース部115と、入力装置120及び出力装置130とのインタフェース装置であるI/Oインタフェース部116と、外部ノードとのデータ通信を制御するインタフェースである通信制御装置117とを備え、それらがバス118を介

して接続された構成である。

【0028】処理装置110は、記録媒体113に記録されたプログラムにしたがって以下に記載する本発明のトランザクション処理システムの処理を実行する。なお、記録媒体113は、磁気ディスク、半導体メモリ、光ディスクあるいはその他の記録媒体であってもよい。

【0029】次に、図1に示したトランザクション処理システムの動作について詳細に説明する。

【0030】本発明のトランザクション処理システムでは、トランザクション処理を実行するコンポーネントを「業務」単位でまとめる。これは以下の考えにしたがって行う。

【0031】一般に、「業務」は複数のトランザクション処理から構成される。例えば、普通預金業務は、入金トランザクション処理、及び出金トランザクション処理等を含んでいる。これらのトランザクション処理では、通常、コンポーネントの呼び出し処理が複数回実行される。

【0032】本発明のトランザクション処理システムでは、このような「業務」に必要な各トランザクション処理に対応する業務フローメソッド231を定義し、複数の業務フローメソッド231を備えた「業務」に対応するクラスを定義し、各クラス毎に業務フロークラス23を作成する。

【0033】本発明のトランザクション処理システムの処理内容は、業務フローの定義処理と実行処理とに分離される。まず業務フローの定義処理について図3及び図4を用いて説明する。

【0034】図3は図1に示したトランザクション処理システムの処理手順を示す図であり、業務フローの定義処理の手順を示すフローチャートである。また、図4は図1に示した業務フロー定義画面の一例を示すイメージ図である。

【0035】図3に示すように、業務フローの定義処理では、ユーザは、サーバノード2により業務フロー定義画面22を起動し（ステップA1）、業務フロー定義画面22の指示にしたがって、まず、業務フローで呼び出す業務コンポーネント24、及びそのメソッドをそれぞれ選択して登録する（ステップA2）。ここで、業務コンポーネント24のメソッドにはそれぞれ複数の引数が設けられているため、各々の引数に所定のキーを指定する（ステップA3）。これらのキーはオブジェクト121に対応するものであり、クライアント11から送信されるメッセージオブジェクト12からオブジェクト121を取り出す際に必要となる。

【0036】続いて、業務コンポーネント24の呼び出し処理の前後で何らかの処理が必要となる場合は、呼び出し前処理／後処理（データ変換規則等）を定義する（ステップA4）。そして、業務フローに挿入すべき業務コンポーネント24がまだあるか否かを判定し、業務

コンポーネント24がある場合はステップA2の処理に戻ってステップA4までの処理を繰り返し、業務フローを作成する（ステップA5）。

【0037】業務フローの作成が完了したら、業務コンポーネント24を呼び出す業務フロークラス23を作成する（ステップA6）。例えば、プログラム言語がJavaの場合、これはクラスファイルとなる。作成した業務フロークラス23はサーバノード2に格納される。

【0038】次に、業務フローの実行処理について図5を参照して説明する。

【0039】図5は図1に示したトランザクション処理システムの処理手順を示す図であり、業務フローの実行処理の手順を示すシーケンス図である。

【0040】図5において、ユーザは、予めクライアント11を用いて業務コンポーネント24に渡すオブジェクト121を作成し、それらをメッセージオブジェクト12にハッシュテーブル形式でクライアントノード1に格納しておく。

【0041】このような状態で、クライアントノード1からサーバノード2にオンライントランザクション処理を依頼する場合、ユーザは、まずクライアント11を用いてサーバノード2にアクセスし、業務フローエンジン21のメソッドを呼び出す。この時、クライアントノード1は、呼び出す業務フロークラス23の名称、及びメッセージオブジェクト12を業務フローエンジン21に引数として渡す。

【0042】サーバノード2は、業務フローエンジン21によって指定された業務フロークラス23をインスタンス化し、対応する業務フローメソッド231を呼び出す。そして、業務フローメソッド231により、業務フロー定義画面22で登録された業務フローにしたがって指定された業務コンポーネント24を順次呼び出し、業務コンポーネント24によるトランザクション処理の結果を業務フローエンジン21を介してクライアント11へそれぞれ返却する。

【0043】なお、業務コンポーネント24を呼び出す際に、業務フロー定義画面22で呼び出し前処理が定義されている場合は、指定された前処理を行った後に業務コンポーネント24を呼び出す。また、呼び出し後処理が定義されている場合は、業務コンポーネント24を実行した後に指定された後処理を行う。クライアントノード1は、クライアント11により業務コンポーネント24を呼び出す時に、業務フローの定義処理の際に設定されたキーにしたがって、メッセージオブジェクト12に格納されたオブジェクト121を業務コンポーネント24に渡していく。

【0044】次に、メッセージオブジェクト12に格納されたオブジェクト121を業務コンポーネント24に渡す手順について図6を用いて説明する。

【0045】図6は図1に示したクライアントノードの

メッセージオブジェクトからサーバノードの業務コンポーネントにオブジェクトを渡すときの手順を示す模式図である。

【0046】図6において、業務フロー定義画面22を用いて業務フローで呼び出す業務コンポーネント24のメソッドを定義するとき、業務コンポーネント24のメソッドの各引数に対応するキーが定義される。図6では、例えば、コンポーネントAのメソッドAの引数arg1に対して、CMPA#MethodA#arg1がキーとして定義されている。

【0047】クライアントノード1は、クライアント11によりメッセージオブジェクト12を作成する時、業務コンポーネント24に渡すオブジェクト121を、上記定義されたキーに基づいてメッセージオブジェクト12に格納する。図6では、例えば、CMPA#MethodA#arg1をキーにしてObj-1がメッセージオブジェクト12に格納される。

【0048】サーバノード2は、業務フロークラス23により業務コンポーネント24を呼び出すとき、メソッドの引数に対して定義されたキーに対応するオブジェクト121をメッセージオブジェクト12から取り出し、引数にセットして業務コンポーネント24のメソッドを呼び出す。図6の場合、例えば、メッセージオブジェクト12からObj-1、Obj-2を取り出し、Method AのパラメータにセットしてComponent AのMethod Aを呼び出している。

【0049】このような方法でメッセージオブジェクト12に含まれる各オブジェクト121を業務コンポーネント24に渡すことで、各オブジェクト121をそれぞれ適切な業務コンポーネント24に渡すことができる。

【0050】次に本発明の形態の効果について説明する。

【0051】上述したように、業務コンポーネントの粒度は任意であるため、業務コンポーネントによる処理単位はユーザにとって分かりにくい場合がある。本発明のように業務フロークラス23を定義することで、トランザクション処理がユーザにとって認識しやすいものとなり、運用管理が容易になる。

【0052】例えば、業務処理の稼働状況を表示する場合を例にして以下に説明する。

【0053】通常、オブジェクト指向トランザクションモニタによる稼働状況の認識単位はコンポーネントであり、オブジェクト指向トランザクションモニタに付属した管理ツール等を用いて確認できる稼働状況は、図7

(a)に示すように、コンポーネントのメソッド呼び出し回数等であり、ユーザにとってわかりにくいものである。

【0054】本発明の場合、例えば、業務フロークラス23の呼び出し回数をカウントして表示すれば、ユーザは、図7(b)に示すように、稼働状況を業務フローク

ラス単位、すなわち業務単位で確認することが可能になる。また、その他物件管理などの各種運用単位を業務毎に行うことができる。

【0055】また、業務や業務フローが単に論理的な単位でなく、クラス、クラスのメソッドというユーザに認識しやすい物理的な実体にマッピングされているため、トランザクション処理プログラムにユーザ独自の処理プログラムを容易に追加することができる。例えば、コンポーネント呼び出し前に独自のロジックを組み込んだり、異常終了時の処理の追加やループの作成等のカスタマイズを容易に行うことができる。また、カスタマイズを業務フロー定義画面を用いて行えるため、カスタマイズがより容易になる。

【0056】さらに、本発明では、オンラインによるトランザクション処理で多数のコンポーネント呼び出し処理が発生する場合でも、クライアント11は業務フローエンジン21のメソッドを呼び出して1つのオブジェクト121を渡す処理だけで済むため、クライアントアプリケーションの簡易化が可能になる。

【0057】次に、本発明の応用例について図面を参照して説明する。

【0058】本発明は主としてコンポーネントベースでトランザクション処理を実行するトランザクション処理システムに関するものであるが、コンポーネントベースでないインターフェースを備えたトランザクション処理システムについても適用可能である。

【0059】例えば、図8に示すように、EAIツール31等を使用してEAIツールに対応するラッパーコンポーネントを業務フローメソッドで呼び出すことにより、コンポーネントベースでないインターフェースを備えたトランザクション処理システムを業務フローの一部に組み込むことができる。また、EAIツール31によりデータをXML形式に変換することでロゼッタネット34への適用も可能である。すなわち、コンポーネントベースではないトランザクション処理システムも業務単位にまとめることが可能になる。

【0060】図8に示すトランザクション処理システムは、図1に示したトランザクション処理システムに、異種システムとの接続処理、あるいはデータ形式変換処理などを行って異種システムとの連携を可能にするEAI(Enterprise Application Integration)ツール31と、EAIツール31をコンポーネントとして取り扱うためのコンポーネントインターフェースを提供するラッパーコンポーネント32と、クライアントノードにインターネットを介して接続されたWebブラウザ33とをさらに有する構成である。その他の構成は図1に示したトランザクション処理システムと同様であるため、その説明は省略する。

【0061】EAIツール31は、メッセージキューイングシステム、ERP(Enterprise Resource Plannin

g)等の異なるインターフェースを有するシステムを簡単に接続するためのツールであり、代表的なものとして、Mercator(Mercator社)、e\*Gate(STC社)等がある。

【0062】また、ラッパーコンポーネント32により異種システムもコンポーネントとして業務フローメソッド231で呼び出すことが可能であり、ユーザは異種システムもトランザクション処理システムに組み込むことが可能になる。

【0063】具体的には、インターネットを介してWebブラウザ33から電子商取引関連の注文がクライアントノードに発注され、クライアントからの依頼によりサーバノードの業務フローメソッドを実行させる。このとき、業務フローメソッドは、自サーバノードに格納された業務コンポーネントを呼び出した後、ラッパーコンポーネント32を呼び出して、EAIツール31によりデータをXML形式に変換し、E-Market Place(ロゼッタネット)に流すようなトランザクション処理が考えられる。このような一連のトランザクション処理を1つの業務として認識することができるため、稼働状況等を業務単位で表示させることができる。

【0064】

【発明の効果】本発明は以上説明したように構成されているので、以下に記載する効果を奏する。

【0065】業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行することで、稼働状況やその他物件管理などの各種運用単位をユーザにとって分かりやすい業務単位で行うことができる。また、業務や業務フローが単に論理的な単位でなく、クラス、クラスのメソッドというユーザに認識しやすい物理的な実体にマッピングされるため、トランザクション処理プログラムにユーザ独自の処理プログラムを容易に追加することができる。

【0066】また、ユーザによるトランザクション処理プログラムのカスタマイズを業務フロー定義画面を用いて行えば、カスタマイズがより容易になる。

【0067】さらに、オンラインによるトランザクション処理で多数のコンポーネント呼出し処理が発生する場合でも、クライアントノードは、サーバノードに格納された業務コンポーネントに、対応する1つのオブジェクトを渡す処理だけで済むため、クライアントアプリケーションの簡易化が可能になる。

【図面の簡単な説明】

【図1】本発明のトランザクション処理システムの一構成例を示すブロック図である。

【図2】図1に示したクライアントノード及びサーバノードの一構成例を示すブロック図である。

【図3】図1に示したトランザクション処理システムの処理手順を示す図であり、業務フローの定義処理の手順を示すフローチャートである。

【図4】図1に示した業務フロー定義画面の一例を示すイメージ図である。

【図5】図1に示したトランザクション処理システムの処理手順を示す図であり、業務フローの実行処理の手順を示すシーケンス図である。

【図6】図1に示したクライアントノードのメッセージオブジェクトからサーバノードの業務コンポーネントにオブジェクトを渡すときの手順を示す模式図である。

【図7】オブジェクト指向トランザクションモニタに付属した管理ツール等を用いて確認できる稼働状況を示す図であり、同図(a)は従来の表示内容を示す表であり、同図(b)は本発明の表示内容を示す表である。

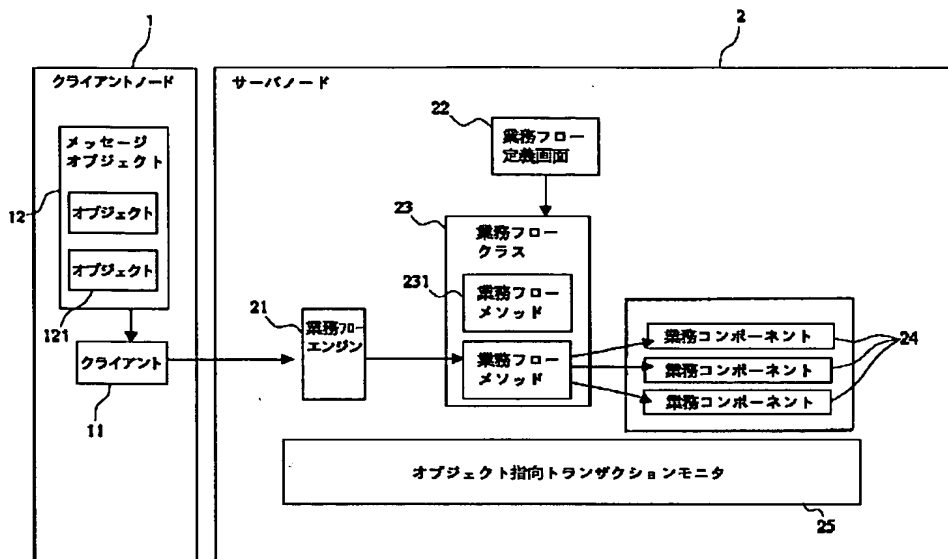
【図8】本発明のトランザクション処理システムの他の構成例を示すブロック図である。

【符号の説明】

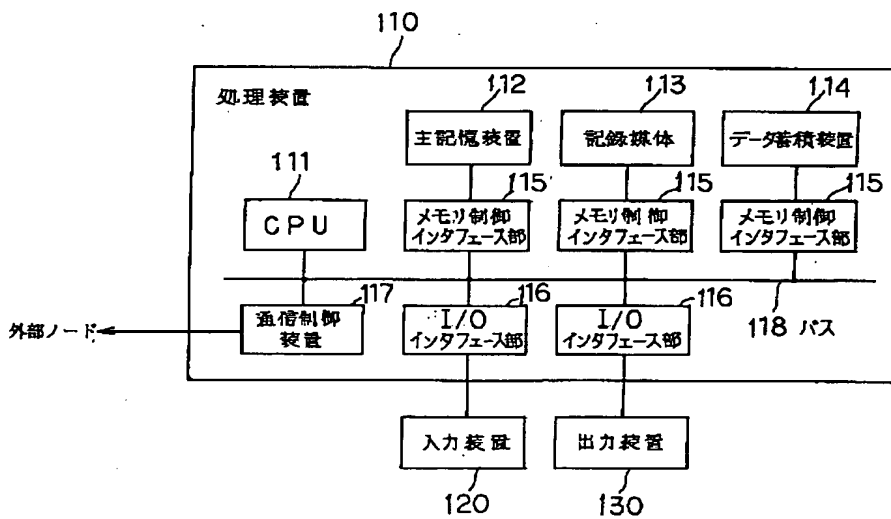
- |     |                     |
|-----|---------------------|
| 1   | クライアントノード           |
| 2   | サーバノード              |
| 11  | クライアント              |
| 12  | メッセージオブジェクト         |
| 21  | 業務フローエンジン           |
| 22  | 業務フロー定義画面           |
| 23  | 業務フロークラス            |
| 24  | 業務コンポーネント           |
| 25  | オブジェクト指向トランザクションモニタ |
| 31  | EAIツール              |
| 32  | ラッパーコンポーネント         |
| 33  | Webブラウザ             |
| 110 | 処理装置                |
| 111 | CPU                 |
| 112 | 主記憶装置               |
| 113 | 記録媒体                |
| 114 | データ蓄積装置             |
| 115 | メモリ制御インタフェース部       |
| 116 | I/Oインタフェース部         |
| 117 | 通信制御装置              |
| 118 | バス                  |
| 120 | 入力装置                |
| 121 | オブジェクト              |
| 130 | 出力装置                |
| 231 | 業務フローメソッド           |



【図1】



【図2】



【図7】

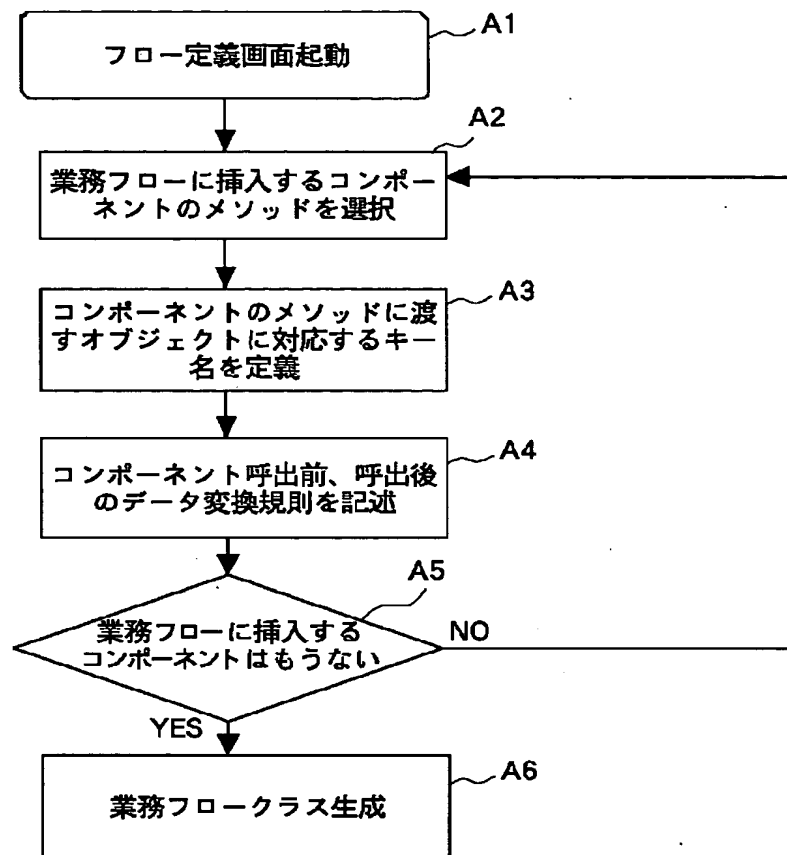
(a)

コンポーネント、メソッド	回数
Comp1, Method1	10回/秒
Comp1, Method2	10回/秒

(b)

コンポーネント、メソッド	回数
普通預金業務	20件/秒
定期預金業務	10回/秒

【図3】



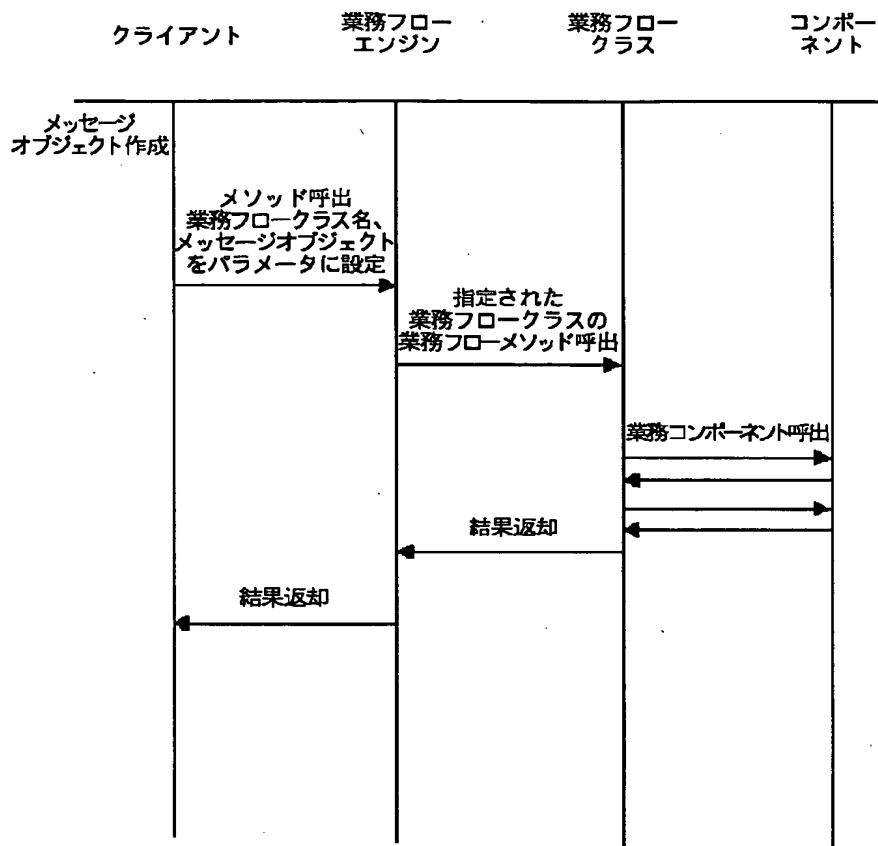
【図4】

<div style="border: 1px solid black; padding: 5px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black; margin-bottom: 5px;"> <span>FlowName</span> <span>flowA</span> </div> <div style="font-family: monospace; font-size: 0.9em;"> <pre> ■ flowA + flowA + flowStart + TX START + ■ EJB1.Method1 + ■ EJB1.Method2 + ■ EJB2.Method1 + TX COMMIT + ■ catch + + Exception + final </pre> </div> <div style="display: flex; justify-content: space-between; border-top: 1px solid black; margin-top: 5px;"> <span>UP</span> <span>DOWN</span> <span>Delete</span> </div> </div>	<div style="border: 1px solid black; padding: 5px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black; margin-bottom: 5px;"> <span>CALL EJB</span> <span>EJB1</span> <span>Method</span> <span>Method1</span> </div> <table border="1" style="width: 100%; border-collapse: collapse; font-size: 0.8em;"> <thead> <tr> <th>Parameter</th> <th>Message Key</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>EJB1.Method1 arg1</td> <td>Object</td> </tr> <tr> <td>2</td> <td>EJB1.Method1 arg2</td> <td>int</td> </tr> </tbody> </table> <div style="display: flex; align-items: center; margin-top: 5px;"> <span>Result</span> <span>Object</span> <span>⇒</span> <span>r1</span> </div> <table border="1" style="width: 100%; border-collapse: collapse; font-size: 0.8em;"> <thead> <tr> <th>Pre Call Condition</th> <th>Next Call Condition</th> </tr> </thead> <tbody> <tr> <td>NONE</td> <td>NONE</td> </tr> <tr> <td></td> <td></td> </tr> </tbody> </table> </div>	Parameter	Message Key	Type	1	EJB1.Method1 arg1	Object	2	EJB1.Method1 arg2	int	Pre Call Condition	Next Call Condition	NONE	NONE		
Parameter	Message Key	Type														
1	EJB1.Method1 arg1	Object														
2	EJB1.Method1 arg2	int														
Pre Call Condition	Next Call Condition															
NONE	NONE															

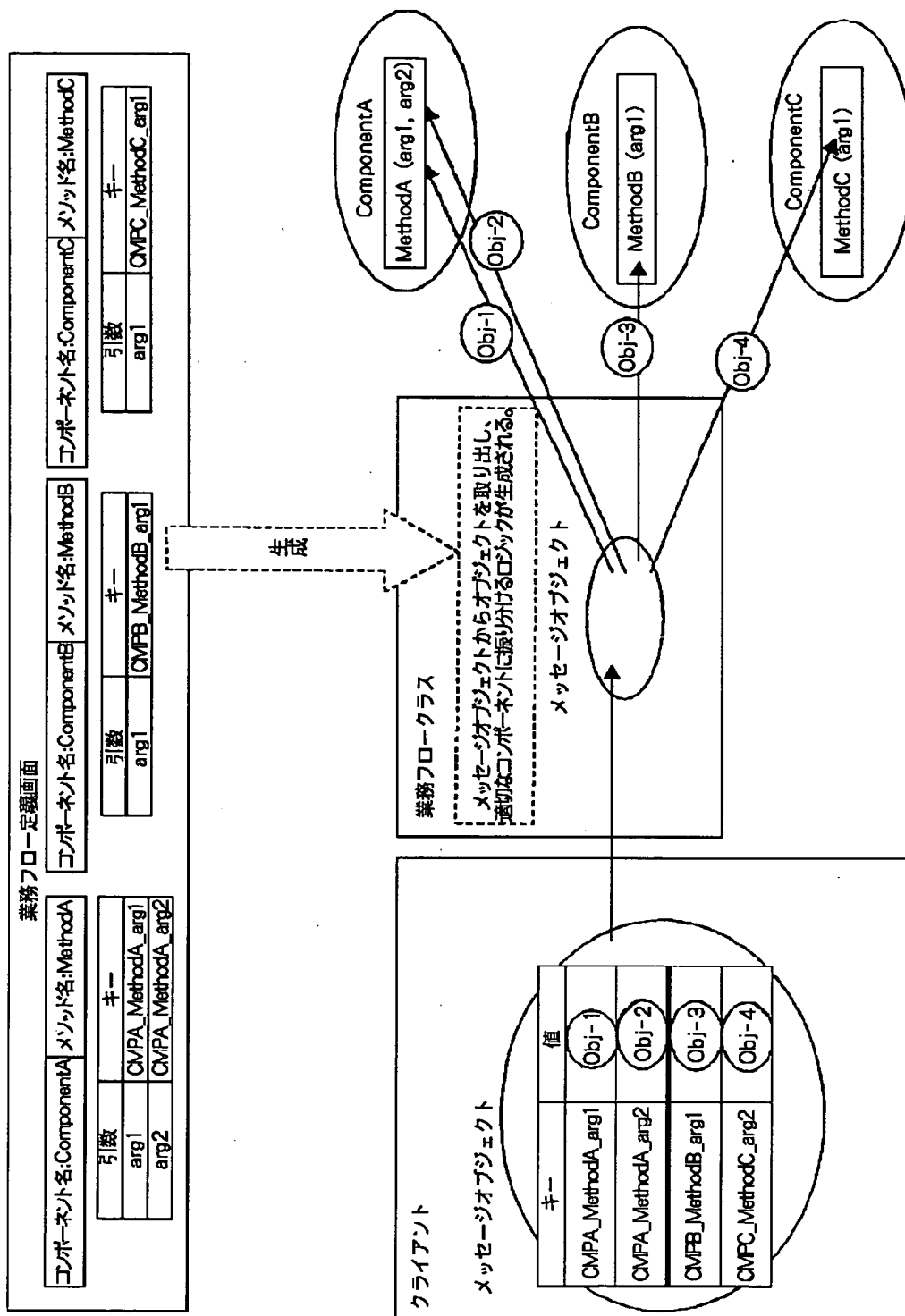
  

<div style="border: 1px solid black; padding: 5px;"> <div style="font-family: monospace; font-size: 0.9em;"> <pre> ■ C: * + LIB1 + EJB1.jar + EJB2.jar + </pre> </div> </div>	<div style="border: 1px solid black; padding: 5px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black; margin-bottom: 5px;"> <span>EJB Name</span> <span>EJB1</span> <span>Version</span> </div> <table border="1" style="width: 100%; border-collapse: collapse; font-size: 0.8em;"> <thead> <tr> <th>Modifier</th> <th>Method</th> <th>Parameter</th> </tr> </thead> <tbody> <tr> <td>Object</td> <td>Method1</td> <td>Object, int</td> </tr> <tr> <td>Object</td> <td>Method2</td> <td>Object, int, String</td> </tr> <tr> <td>String</td> <td>Method3</td> <td>Object</td> </tr> </tbody> </table> <div style="display: flex; justify-content: space-between; border-top: 1px solid black; margin-top: 5px;"> <span>Add</span> <span>Close</span> </div> </div>	Modifier	Method	Parameter	Object	Method1	Object, int	Object	Method2	Object, int, String	String	Method3	Object
Modifier	Method	Parameter											
Object	Method1	Object, int											
Object	Method2	Object, int, String											
String	Method3	Object											

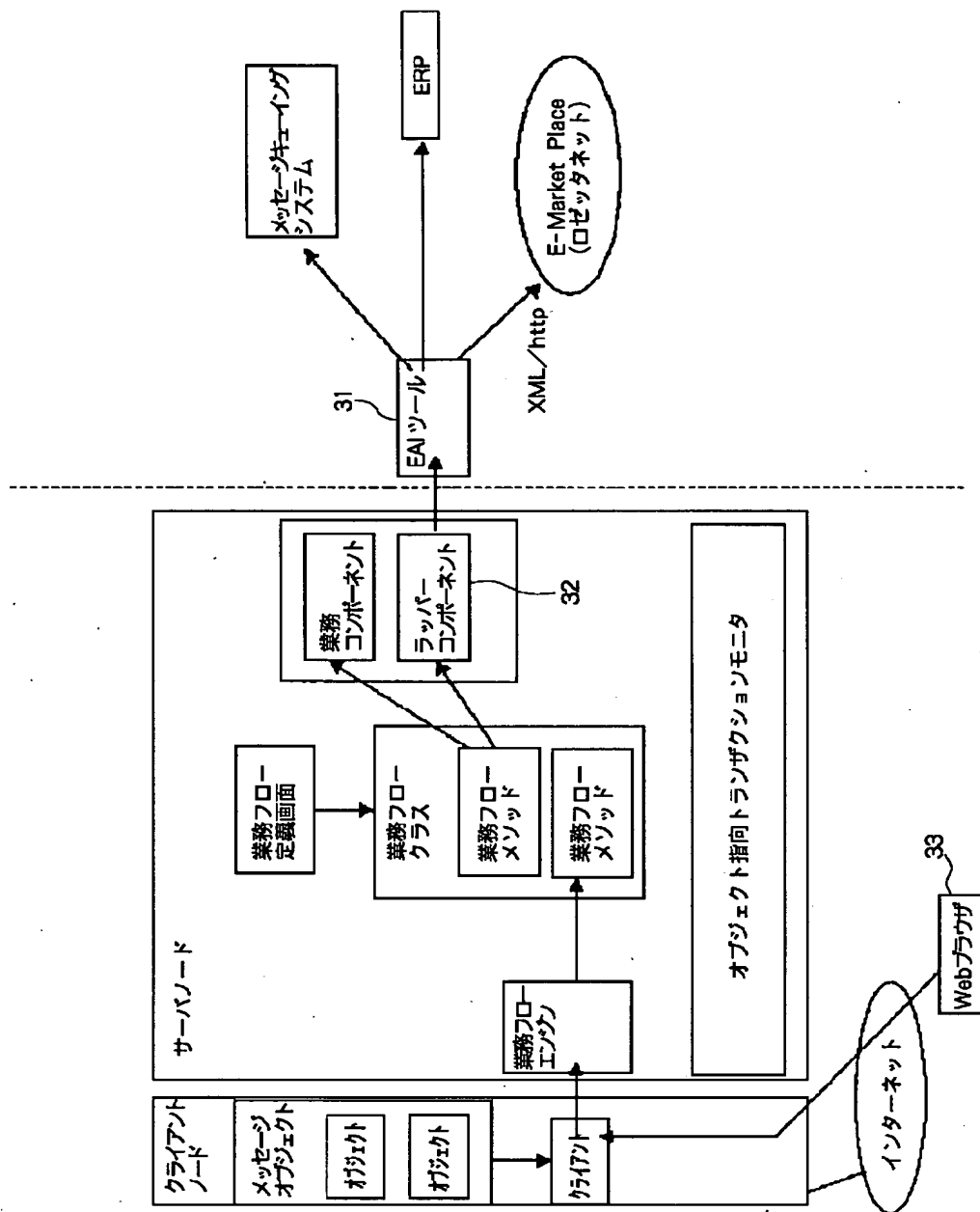
【図 5】



【図6】



【図8】



フロントページの続き

(51) Int. Cl.<sup>7</sup>

G 0 6 F 15/16

識別記号

6 2 0

F I

G 0 6 F 9/06

ターコード (参考)

6 2 0 A